Corvus: Uma solução *Sandbox* e de *Threat Intelligence* para Identificação e Análise de *Malware*

Marcus Botacin[1], **Fabricio Ceschin**[1], **André Grégio**[1]

[1] Universidade Federal do Paraná

{mfbotacin, fjoceschin, gregio}@inf.ufbr.pr

Abstract. *In this paper, we introduce* `Corvus`*, a web platform for identifying, triaging, and analyzing malware threats. By using it, users can submit unknown PDF and/or Android, Linux, and Windows files to static, dynamic, and data analysis procedures. The obtained data is reported in tables, dynamic graphs, and can be exported in formats suitable for the usage by security incident response teams. Available at* `https://corvus.inf.ufpr.br/`

Resumo. *Neste artigo, apresentamos* `Corvus`*, uma solução web para a identificação, triagem e análise de ameaças. Através da plataforma, usuários podem submeter arquivos PDF e executáveis Android, Linux e Windows desconhecidos para procedimentos de análise estática, dinâmica e de análise de dados. Os dados obtidos são reportados através de tabelas, gráficos dinâmicos e podem ser exportados em formatos adequados para a utilização por equipes de resposta à incidentes de segurança. Disponível em* `https://corvus.inf.ufpr.br/`

1. Motivação

Software Malicioso (*malware*) é uma ameaça constante à segurança dos sistemas computacionais e diversas soluções de defesa tem sido desenvolvidas para tentar lidar com este problema. De maneira geral, estas soluções podem ser divididas entre as diretamente aplicadas na máquina do usuário (*endpoint*) e as aplicadas à nível das entidades que fornecem serviços de inteligência e gestão. No primeiro caso, destacam-se soluções Antivírus (AVs) e *firewall*. No segundo caso, destacam-se soluções de *sandbox* e de inteligência de ameças (*threat intelligence*).

As soluções *sandbox* permitem a execução de artefatos até então desconhecidos em um ambiente controlado, permitindo assim a caracterização destes e a eventual identificação e classificação como um artefato malicioso. As soluções de inteligência de ameaças operam sobre estes dados de modo a permitir o levantamento de informações sobre a ameaça encontrada em um escopo mais elevado. A combinação das duas soluções pode permitir, por exemplo, a atribuição dos ataques a um determinado grupo, a identificação de artefatos semelhantes, ou mesmo a previsão de movimentos futuros do atacante através da inspeção dos *hosts* contactados.

A importância destes tipos de soluções já foi amplamente reconhecido pela academia e pela indústria, que fornece soluções notáveis, como os serviços Virustotal[1], JoeBox[2] e Hybrid Analysis[3]. Contudo, tais soluções operam de forma global, não dando conta das

[1] `https://www.virustotal.com/gui/`
[2] `https://www.joesecurity.org/`
[3] `https://www.hybrid-analysis.com/`

especificidades das ameaças presentes na Internet brasileira (e.g., tipos de arquivos não usuais [Botacin et al. 2021]).

Acreditamos que a Internet brasileira se beneficiaria enormemente de uma solução deste tipo que pudesse ser alimentada pela comunidade local. Para este propósito, propomos a plataforma *Corvus* (`https://corvus.inf.ufpr.br/`), apresentada por meio deste artigo. Nosso objetivo ao publicizar a plataforma é de trazer atenção para o tema da cibersegurança da Internet nacional e apontar caminhos futuros para outros pesquisadores interessados no assunto.

A seguir, discutiremos como `Corvus` é implementado (Seção 2), como pode ser utilizado (Seção 3), bem como apresentamos alguns resultados preliminares (Seção 4).

2. Implementação

A maior vantagem possibilitada pela plataforma *Corvus* é a integração de múltiplas ferramentas de análise em uma única solução. Para tanto, desenvolvemos uma arquitetura de tratamento de todas as etapas de um processo de análise automatizado de *malware*. A arquitetura proposta é exibida na Figura 1.

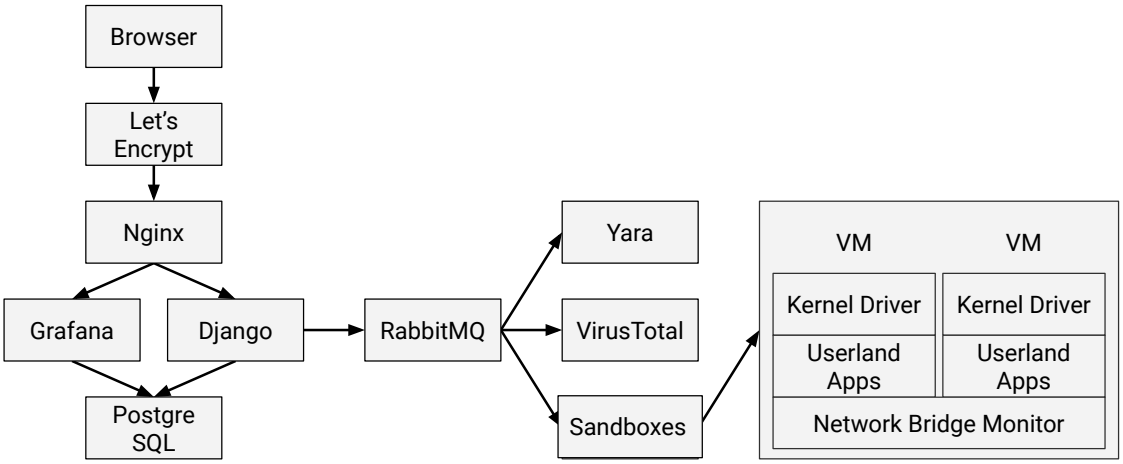

Figura 1. Arquitetura da plataforma *Corvus*.

Quando um usuário faz uma solicitação ao sistema *Corvus* através da *web*, o navegador do usuário se comunica de forma criptografada (via `Let's Encrypt`) com o *frontend* da plataforma (servidor `Nginx` atuando como *proxy* reverso). Se uma requisição para obtenção de dados de análise é identificada, o `Nginx` redireciona a requisição para a solução de visualização de dados `Grafana`, que, por sua vez, obtém os dados através da comunicação com o servidor de banco de dados (`Postgres` atuando em modo `NoSQL`). Caso o usuário deseje submeter um novo artefato para análise, o Nginx direcionará a requisição para um serviço *web* baseado no *framework* `Django`, que atua como um *middleware*, provendo um conjunto de APIs para uso dos módulos internos e externos. Todas as requisições de análise são adicionadas à uma fila (gerenciada por `RabbitMQ`) e entregues aos módulos de análise em *backend*. As atividades no *backend* são divididas entre tarefas de análise estática e dinâmica. Os resultados são submetidos de volta ao servidor através das APIs do `Django` para serem armazenados no banco de dados e serem consultados por requisições posteriores.

2.1. *Frontend*

O *frontend* é composto por todos os componentes *web* responsáveis pela coleta dos *payloads* submetidos pelos usuários e pela exibição dos resultados de análise. Ele oferece todas as facilidades básicas para a manipulação dos resultados de análise, incluindo as capacidades de busca, agrupamento e visualização. Em particular, este possui um mecanismo de construção de grafos dinâmicos a partir dos resultados de análise. Além disso, existe um sistema de controle de acesso que permite gerenciar usuários e suas respectivas chaves de API, além de criar submissões privadas em que somente o usuário tem acesso ao relatório de um arquivo submetido.

2.2. *Middleware*

O *middleware* é composto pelo conjunto de componentes *web* e bancos de dados que proveem APIs a serem utilizadas pelo usuários e pelos componentes internos da plataforma[4]. Em nossa implementação, todos os módulos internos se comunicam indiretamente com o usuário através do *middleware*, de modo que todos os resultados de análise são armazenados no banco de dados através de chamadas de API. Esta construção modular permite que diferentes módulos se comuniquem em paralelo e que novos métodos de análise sejam incorporados sem a interrupção do sistema para a refatoração de código.

2.3. Análise Estática

O módulo interno de análise estática é composto por múltiplas ferramentas que operam de forma independente. Ao executar cada solução em uma *thread* diferente evitamos o risco de um *crash* em uma dada solução (muito comum ao análisar códigos maliciosos) comprometer todos os resultados de análise. Todas as *threads* são alimentadas pela mesma fila, de modo que novos arquivos a serem analisados são entregues a todas as ferramentas. Conforme ocorre a finalização da execução das ferramentas, os resultados são enviados ao banco de dados. A correlação entre os dados é realizada pelo *frontend* e não pelas ferramentas individualmente. Atualmente, *Corvus* suporta as seguintes tarefas de análise estática: (i) escaneamento por soluções antivírus; (ii) casamento de padrões com assinaturas já conhecidas; (iii) desmontagem do binário; (iv) extração de arquivos embutidos; e (v) análise dos campos do cabeçalho do binário.

2.4. Análise Dinâmica e *Sandbox*

O módulo de análise dinâmica é o principal componente de análise da nossa plataforma, visto que este tipo de inspeção permite a descoberta de comportamentos ocultos por processos de ofuscação. Todo arquivo suspeito submetido à plataforma é executado na solução de *sandbox*, que captura a atividade de processos do sistema, no sistema de arquivos e na hierarquia de registros. A solução executa aplicações em nível de usuário e monitora o sistema através de um *driver* de *kernel*, cuja implementação foi descrita em um artigo anterior [Botacin et al. 2018]. A plataforma implementa, ainda, um mecanismo de provisionamento sob demanda de máquinas virtuais (VMs) de análise. Este mecanismo é também responsável por coletar informações de tráfego de rede nos adaptadores virtuais de cada VM. Diferentemente do módulo de análise estática, dado a natureza do procedimento, o módulo dinâmico correlaciona nativamente os resultados de análise. Por exemplo, informações de tráfego de rede extraídas do traço de execução são *parseadas* pelo módulo de análise de tráfego de rede para complementação das análises.

[4] Consulte a documentação das APIs em `https://corvus.inf.ufpr.br/docs/`

3. Aplicação

A plataforma *Corvus* aceita a submissão de arquivos executáveis de Windows (PE), Linux (ELF), Android (APK), bem como arquivos PDF. Embora cada um destes formatos seja analisado por módulos específicos, os resultados são apresentados de forma combinada em uma interface uniforme. Um relatório diferente é produzido para cada arquivo submetido. Diferentes relatórios para o mesmo arquivo podem ser produzidos se requisições de novas análises são solicitadas. Este tipo de reanálise é útil para a identificação de novos *payloads* remotos ou a confirmação de que servidores C&C foram retirados do ar. Os dados presentes nos relatórios podem ser exportados em formatos adequados para o uso por equipes de resposta à incidentes (CSIRT), como o formato MISP[5] e STIX[6].

Os relatórios contém informações de análise estática, dinâmica, tráfego de rede, detecção por soluções antivírus e por classificadores baseados em aprendizado de máquina. O relatório de análise estática exibe indicadores de comprometimento (IoCs) baseados nas *strings* extraídas dos binários, em *matches* de regras `yara` (Figura 2), no *disassembly* do binário e no *parsing* dos cabeçalhos dos binários executáveis. Adicionalmente, exibe-se também potenciais arquivos embutidos (*malware* do tipo *dropper*) caso estes sejam identificados.

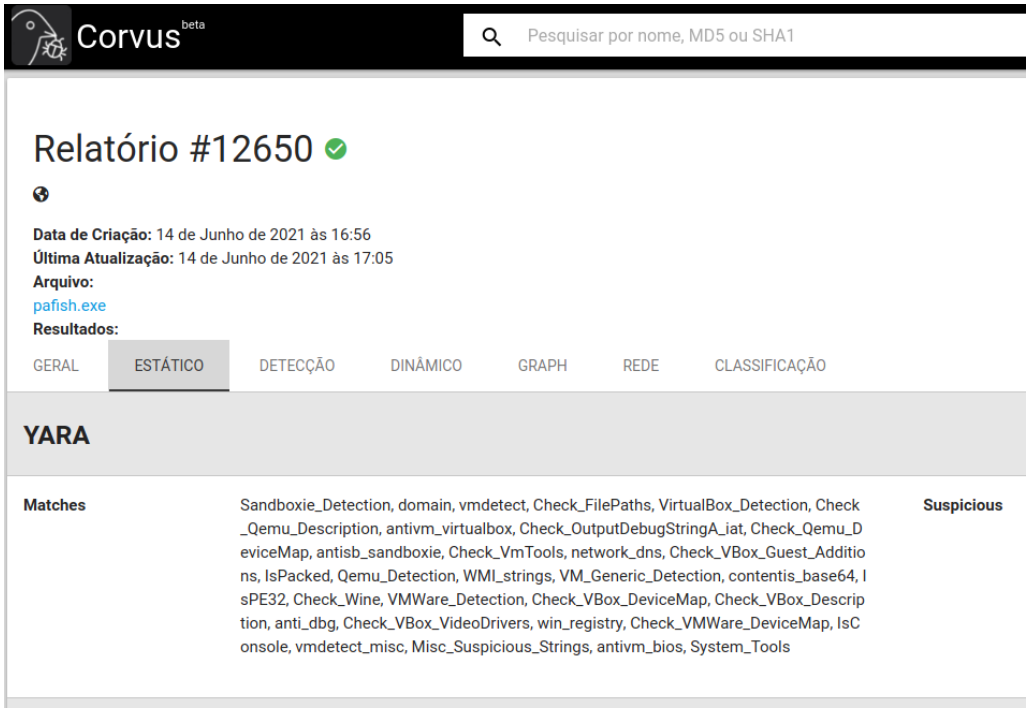

Figura 2. Relatório de Análise Estática.

Os relatórios de análise dinâmica apresentam o traço de execução do binário dentro da solução *sandbox*. Isto inclui, para o ambiente Windows, os arquivos acessados (Figura 3), os processos criados, e as chaves de registro modificadas. No ambiente Linux, os dados incluem as chamadas de sistema e de bibliotecas.

A análise do tráfego de rede exibe todo o tráfego TCP e UDP relacionado a execução do exemplar. Além da identificação de endereço IP e porta, identifica-se potenciais pro-

[5] `https://www.misp-project.org/` [6] `https://stixproject.github.io/`

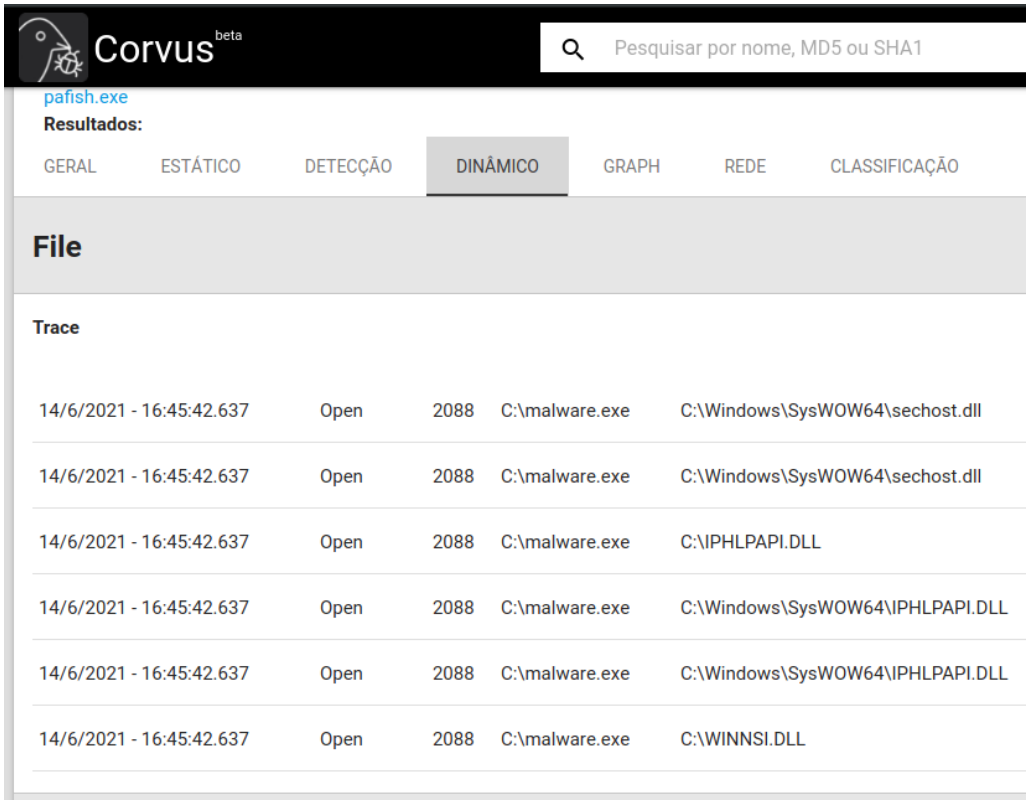

Figura 3. Relatório de Análise Dinâmica.

tocolos utilizados. Os protocolos HTTP e DNS (Figura 4) são especialmente *parseados* tanto em relação a requisições quanto a respostas para exibir informações comumente associadas a execução de *malware*, como o *download* de *payloads* maliciosos externos.

O relatório de classificação por mecanismos de aprendizado de máquina inclui diversos modelos apresentados na literatura de deteccção de malware para Windows. Dentre eles, os modelos EMBER [Anderson and Roth 2018] (LightGDM), MalConv [Raff et al. 2017], Non-Negative MalConv [Fleshman et al. 2019] e variações do modelo Need for Speed [Ceschin et al. 2018], de nossa autoria, treinado com diferentes conjuntos de dados e classificadores. Conforme apresentado na Figura 5, o *Corvus* exibe os resultados para cada modelo, incluindo o rótulo dado ao arquivo (suspeito ou não suspeito) e o nível de confiança da decisão tomada por cada modelo.

A plataforma *Corvus* também suporta funcionalidades de segurança ofensiva de modo a permitir testes de robustez dos mecanismos de segurança por equipes de defesa. Em especial, *Corvus* possui um mecanismo automático de geração de exemplares evasivos através do *embedding* de um binário original em um novo binário do tipo *dropper*. Ao ser acionado, este mecanismo gera uma variante do binário originalmente analisado, permitindo tanto a verificação da evasão dos classificadores presentes na plataforma quanto a preservação das funcionalidades originais através dos IoCs dinâmicos.

Os relatórios podem ser visualizados individualmente ou agrupados por critérios definidos pelo usuário (coleção de arquivos). Pode-se, por exemplo, utilizar a similaridade dos binários como critério, de modo a se criar uma coleção de variantes de *malware*. Pode-

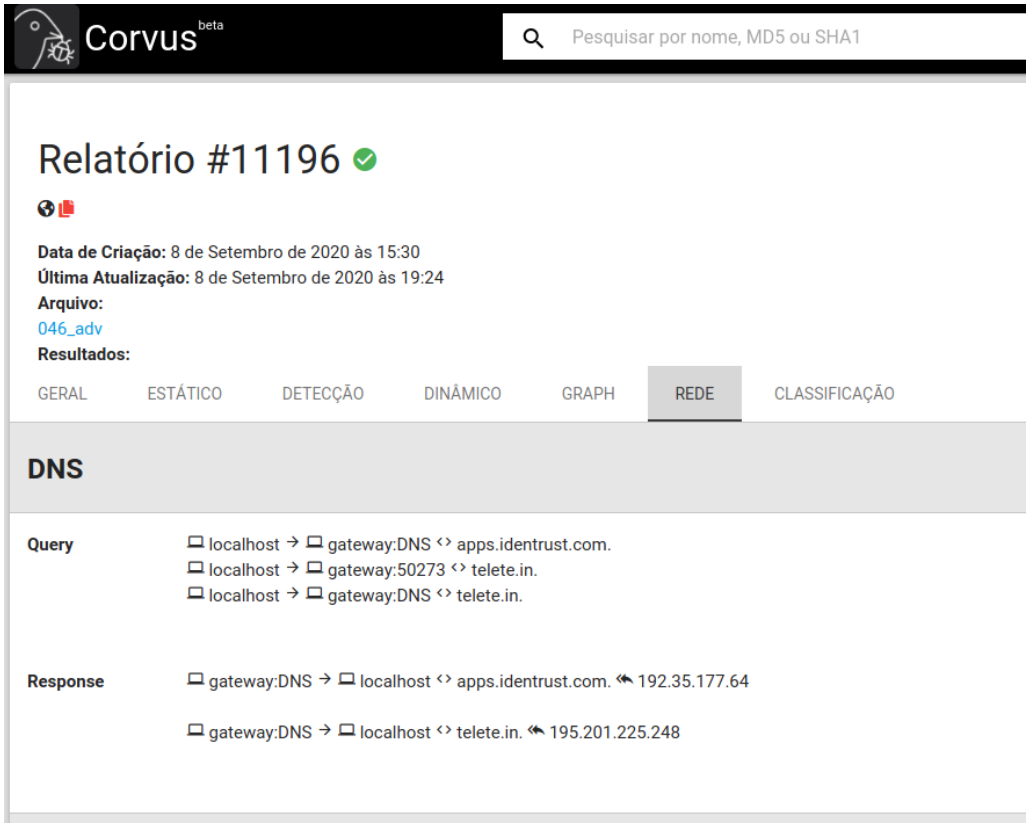

Figura 4. Relatório de Análise do Tráfego de Rede.

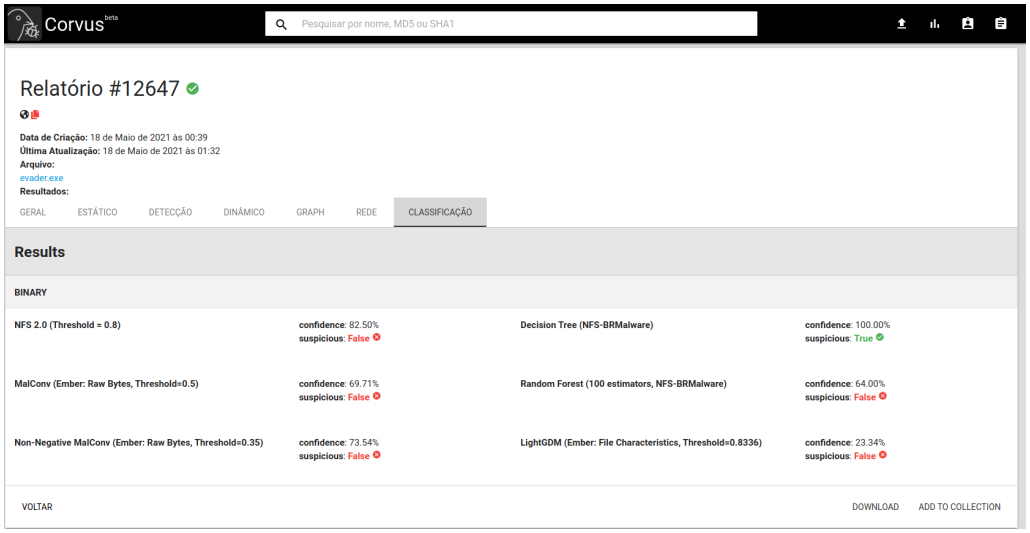

Figura 5. Relatório de Análise por Aprendizado de Máquina.

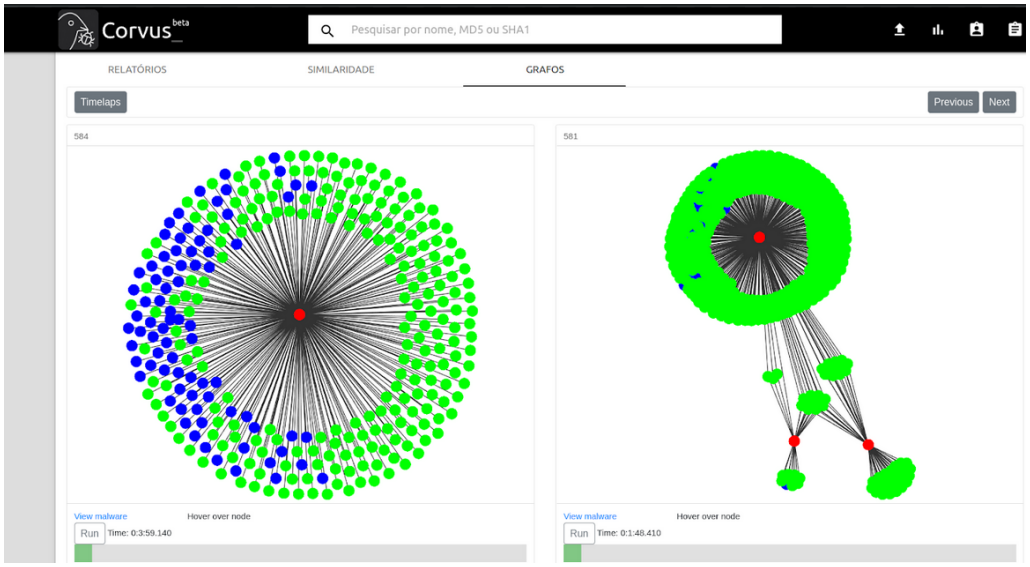

Figura 6. Visualização comparativa de 2 traços de execução.

se, ainda, plotar as diferenças e similaridades entre os traços dinâmicos dos exemplares presentes em uma coleção (Figura 6). Isto permite visualizar, por exemplo, a similaridade entre um binário original e sua versão *dropper*.

4. Conclusão

Neste artigo, apresentamos `Corvus`, uma solução *web* para o tratamento de arquivos maliciosos. Descrevemos a implementação e a operação do sistema, de modo que qualquer usuário possa se beneficiar dos recursos já existentes bem como contribuir com a implementação de novas capacidades.

Desde sua criação, `Corvus` foi utilizado para o desenvolvimento de múltiplos trabalhos de pesquisa. Além disso, muitas das ferramentas desenvolvidas no ambiente de pesquisa foram integradas como recursos na solução. A saber, o seguintes artigos publicados estão diretamente relacionados à plataforma desenvolvida:

- Uma análise do *malware* ativo na Internet brasileira [Botacin et al. 2021]
- Um decompilador de *malware* para Linux [Botacin et al. 2019]
- Modelos de classificação de arquivos baseados em aprendizado de máquina [Ceschin et al. 2018]
- Um gerador de exemplares adversários para evadir classificadores [Ceschin et al. 2020]

As ferramentas que compõem o sistema `Corvus` tem seu código fonte disponível na Internet. A saber:

- Uma das versões da *sandbox* está disponível em `https://github.com/marcusbotacin/BranchMonitoringProject`
- O classificador de arquivos Windows está disponível em `https://github.com/fabriciojoc/mlsec2020-needforspeed`

- O classificador de arquivos ELF está disponível em `https://github.com/marcusbotacin/ELF.Classifier`
- O gerador de exemplares evasivos está disponível em `https://github.com/marcusbotacin/Dropper`
- Os detectores de evasão em código *assembly* estão disponíveis em `https://github.com/marcusbotacin/Anti.Analysis`
- O descompilador de executáveis está disponível em `https://github.com/marcusbotacin/Reverse.Engineering.Engine`

Além disso, diferentes soluções resultantes de projetos de pesquisa são frequentemente integradas à plataforma de acordo com a evolução das mesmas.

Referências

Anderson, H. S. and Roth, P. (2018). EMBER: An Open Dataset for Training Static PE Malware Machine Learning Models. *ArXiv e-prints*.

Botacin, M., Aghakhani, H., Ortolani, S., Kruegel, C., Vigna, G., Oliveira, D., Geus, P. L. D., and Grégio, A. (2021). One size does not fit all: A longitudinal analysis of brazilian financial malware. *ACM Trans. Priv. Secur.*, 24(2).

Botacin, M., de Geus, P. L., and Grégio, A. (2018). The other guys: automated analysis of marginalized malware. *Journal of Computer Virology and Hacking Techniques*, 14(1):87–98.

Botacin, M., Galante, L., de Geus, P., and Grégio, A. (2019). Revenge is a dish served cold: Debug-oriented malware decompilation and reassembly. In *Proceedings of the 3rd Reversing and Offensive-Oriented Trends Symposium*, ROOTS'19, Vienna, Austria. Association for Computing Machinery.

Ceschin, F., Botacin, M., Lüders, G., Gomes, H. M., Oliveira, L., and Gregio, A. (2020). No need to teach new tricks to old malware: Winning an evasion challenge with xor-based adversarial samples. In *Reversing and Offensive-Oriented Trends Symposium*, ROOTS'20, page 13–22, Vienna, Austria. Association for Computing Machinery.

Ceschin, F., Pinage, F., Castilho, M., Menotti, D., Oliveira, L. S., and Gregio, A. (2018). The need for speed: An analysis of brazilian malware classifiers. *IEEE Security Privacy*, 16(6):31–41.

Fleshman, W., Raff, E., Sylvester, J., Forsyth, S., and McLean, M. (2019). Non-negative networks against adversarial attacks.

Raff, E., Barker, J., Sylvester, J., Brandon, R., Catanzaro, B., and Nicholas, C. (2017). Malware detection by eating a whole exe.